

Manuscript version: Author's Accepted Manuscript

The version presented in WRAP is the author's accepted manuscript and may differ from the published version or Version of Record.

Persistent WRAP URL:

<http://wrap.warwick.ac.uk/129175>

How to cite:

Please refer to published version for the most recent bibliographic citation information. If a published version is known of, the repository item page linked to above, will contain details on accessing it.

Copyright and reuse:

The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions.

Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

Publisher's statement:

Please refer to the repository item page, publisher's statement section, for further information.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk.

Self-Taught Learning of the Accessible Surface Area of Proteins

*Fahad ul Hassan, Dr. Fayyaz ul Amir Afsar Minhas[†]

*D.I. Khan Institute of Nuclear Medicine and Radiotherapy, D.I.Khan
Email: fahadalhassan1@gmail.com

[†]Department of Computer and Information Sciences
Pakistan Institute of Engineering and Applied Sciences, Nilore, Islamabad
Email: afsar@pieas.edu.pk

Abstract—In this work, We have investigated self-taught learning methods along with deep neural network to predict one very important property of proteins from protein sequences, i.e., accessible surface area and to solve the problem of feature selection using a comparatively new machine learning concept i.e self-taught learning. Accessible surface area (ASA) is predicted using Support Vector Regression (SVR). Two SVR optimization methods with three different types of features including position dependent k-spectrum, Blosum 62 substitution matrix and position specific scoring matrix (PSSM) are used separately to predict ASA. Optimization methods used are quadratic programming (QP) solver to solve dual form of SVR with Radial Basis Function (RBF) kernel and stochastic sub-gradient optimization (SSGO) algorithm to optimize a linear SVR. QP based solver produced better results with PSSM features as compared to SSGO algorithm but SSGO algorithm is significantly time efficient as compared to first method. Two self-taught learning algorithms are used to learn better representations of data which are Sparse autoencoder and dictionary learning. These algorithms are trained using unlabeled data extracted from non-redundant protein database. Labeled examples are transformed into new representations using trained algorithms and SSGO based regressor is used to predict ASA. We have used these algorithms with position dependent k-spectrum and Blosum 62 substitution matrix based features as these are easier to compute as compared to PSSM which uses a lot of data and time. We got comparable results with these algorithms as those of using simple data without using these algorithms. A feed forward deep neural network having three layers has been used for comparison purposes. This model is trained and tested using labeled data. It produced better results than self-taught learning algorithms with Blosum features.

Keywords—*Accessible Surface Area, Self-Taught Learning, Sparse Autoencoder, Dictionary Learning, Stochastic Subgradient Optimization based Regressor*

I. INTRODUCTION

ASA of residues in a protein sequence is important because it allows prediction of many other properties of proteins like secondary structure, residue depth, hydrophobicity, trans-membrane topology, and binding associated conformational changes. ASA is also a good measure of folding state of a protein. Because active sites of proteins are often located at their surface, the prediction of exposed residues is important for understanding the conformational changes of proteins upon binding. Marsh et al. [1] have shown that ASA of buried residues in protein structure is associated with protein flexibility and it can be used to predict protein flexibility. Liu et

al. [2] have suggested the use of ASA and residue depth along with other sequence and structure based features can improve the sensitivity and accuracy of fold recognition of proteins. Xia et al. [3] have used solvent accessibility to predict hot spots in protein structure. Hot spots are residues which have major contribution of binding free energy of protein interactions.

There are many methods to predict solvent accessibility of proteins. Earlier methods treat solvent accessibility as multi-state classification problem and solved it using SVM [4], two stage SVM [5], neural networks [6] and bidirectional recurrent neural networks [7].

For the first time Ahmad et al. [8] have treated solvent accessibility prediction as regression problem because real value of ASA gives more meaningful information than to classify the residues as buried and exposed types. Real value of ASA gives direct measure of area of a residue exposed to surrounding solvent. Later on, several other methods have been proposed to predict continuous real value of ASA using different features which include support vector regression using neighborhood information [9] [10], feed forward and recurrent neural networks using PSSM based features [11] and neural network based method using evolutionary information in the form of multiple sequence alignment [12].

Recently, machine learning strategies has been proposed to predict multiple properties of proteins simultaneously. Adamczak et al. [13] have predicted secondary structure and solvent accessibility simultaneously. Deep neural networks have been used for prediction of local structural properties of proteins. Deep neural networks are artificial neural networks with multiple layers. Qi et al. [14] proposed unified multi-task architecture of deep neural networks using PSSM and neighborhood information to predict multiple local properties of proteins including solvent accessibility but they treated it as classification problem. Recently Heffernan et al. [15] developed a method using iterative deep neural network to predict secondary structure, solvent accessibility and torsional angles. They used PSSM based features as input to neural network. These methods have improved prediction results using deep neural networks but they have not taken care of timing complexity as it is time consuming task to compute PSSM.

To tackle above mentioned problems, we have used simple features based on neighborhood information and Blosum 62 substitution matrix extracted from unlabeled data to train self-

taught learning [16] algorithms including sparse autoencoder [17], dictionary learning algorithm [18] and PCA. These methods give better feature representations which are used as input to stochastic sub-gradient based regressor to predict ASA. Again stochastic regressor is chosen keeping in view its timing efficiency.

II. MATERIALS & METHODS

We have used two optimization methods to solve SVR problem to predict ASA of residues. First one is quadratic programming solver to solve dual form of SVR and second one is stochastic sub-gradient optimization algorithm. We have used simple features based on neighborhood information and Blosom62 substitution matrix extracted from unlabeled data to train self-taught learning [16] algorithms including sparse autoencoder [17], dictionary learning algorithm [18] and PCA. These methods give better feature representations which are used as input to stochastic sub-gradient based regressor to predict ASA. Again stochastic regressor is chosen keeping in view its timing efficiency. Deep neural network has also been used for comparison purposes.

A. Dataset and Preprocessing

Protein-Protein Docking Benchmark 4.0 [22] is selected to get protein sequences used for feature extraction. This dataset has separate PDB files for both bound and unbound structure of same protein. Also separate files for ligand and receptor proteins are available. We have used 343 protein sequences of unbound state for ASA prediction. STRIDE tool is used to compute ASA which are used as target values [23]. All protein sequences shorter than 30 amino acids in length are removed from dataset as they don't provide enough neighborhood information to feature set. Proteins having pairwise sequence similarity of more than 30% are excluded using CD hit tool. Some PDB files contain more than one chains. Those chains are separated and each PDB file input to STRIDE contains single chain as presence of second chain affects the ASA area. Before ASA calculation using Stride PDB files are cleaned using Pymol to remove HETATM (heterogeneous atoms) which are not part of protein sequence. Also some PDB files contain residues other than 20 standard amino acids called as unclassified residues. These residues are removed as presence of HETATM and unclassified residues pose errors while computing ASA using STRIDE. To perform 10-fold cross-validation, we divided the dataset into ten groups each containing equal number of data and target values. One group was chosen as testing set while nine groups are merged to form training set. To measure the performance of SVR in this application, Pearsons correlation coefficient between predicted and observed RASA values were calculated. The Pearsons correlation coefficient is defined as the ratio of the covariance between the predicted and observed ASA values per residue to the product of the standard deviations.

B. Feature Extraction

The feature vector x_i representing a residue is extracted by a sliding window method using single protein sequence as input. The window size is set at 21 residues. Increasing the window size can provide more local information. It is reasonable to expect that prediction accuracy would increase

with the enlargement of the window size. However, we found that window size has a very limited effect on prediction accuracy. The ASA value is computed for the central residue. In the sliding window, each residue is coded by a 20-dimensional vector, representing the 20 types of amino acids. Therefore, a residue is represented by a $(20 \times 21) = 420$ -dimensional vector. The absolute ASA value for each residue is obtained using Stride. Stride takes a PDB file as input and returns residue location in protein sequence, its torsional angles, secondary structure elements and ASA. RASA values are obtained by Normalizing ASA values by dividing with value of ASA obtained by extended Ala-X-Ala conformation where X is the corresponding residue. This value is used as Alanine is the residue with shortest chain among all amino acids. So surrounding of an amino acid by Alanine will give almost maximum value of ASA. It is important to note that the normalization step can simplify the handling of data, as the ASA values of different amino acids are at the same scale. The SVR algorithm is trained on RASA values and, therefore, predicted values also will be RASA values which can be converted into ASA values by multiplying with corresponding normalizing values.

Three different data representations are used to compare their effects on prediction of ASA values.

- First representation is a binary vector with presence of a particular residue at particular location in window of 21 residues marked as 1 and its absence marked as 0. It is totally based on neighborhood information of a residue.
- In second method each residue in window is represented by corresponding row in Blosom 62 matrix. Blosom 62 is a 20×20 dimensional substitution matrix for sequence alignment in which score for substitution of one amino acid in place of another amino acid is given. Similar amino acids have higher score as compared to dissimilar ones. Obtained feature vectors are normalized to have unit norm.
- In third method position specific scoring matrix is computed for each protein sequence. Size of PSSM is $20 \times l$ where l is the length of protein sequence. A window of 21 residues from this PSSM is picked and normalized to unit norm to be used as feature. PSSM is obtained by performing Multiple Sequence Alignment. Frequency count of each residue at particular location is obtained and logarithm of its frequency with respect to background frequency is computed. It is a measure of presence of a specific amino acid at specific location. PSSM features are better features but it took longer time to perform MSA and compute these features.

C. Sparse Autoencoder

Sparse autoencoder [17] is one of the approaches to use unlabeled data to learn important features automatically. It has outperformed best sophisticated hand-engineered features representations in case of images, audio and text data. It is implemented using feed forward neural network. It tries to learn a function approximate to identity function which gives similar value of output as that of input. A sparsity constraint is

imposed to limit the number of active hidden nodes to reduce redundancy in data to learn important structures in data.

There is no specific rule about number of hidden units but mostly for image classification task number of hidden units are less than input units i.e. to learn compressed representation of data similar to PCA. Even if number of hidden units are larger than number of input units, presence of sparsity constraint will try to eliminate the redundant units and will find improved representation of data. Sparse autoencoder will learn a set of basis vectors ϕ_i such that we can represent input vector x as linear combination of those basis vectors.

$$x = \sum_{i=1}^k a_i \phi_i \quad (1)$$

Then vector of activations a_i 's is used as new feature vector. Cost function for sparse autoencoder consists of three terms given unlabeled data $x_u^{(1)}, x_u^{(2)}, \dots, x_u^{(k)}$ is as follows:

$$f(W, b; x_u) = \frac{1}{k} \sum_i \|x_u^{(i)} - h_{W,b}(x_u^{(i)})\|^2 + \lambda_w \sum_j \|w_j\|^2 + \lambda_{sparsity} \sum_j KL(p\|p_j) \quad (2)$$

First term is the average of the squares of reconstruction error where $h_{W,b}$ is logistic activation function. The second one is regularization term which will decrease the magnitude of weights and prevents overfitting. λ_w is regularization parameter to control relative importance of weight decay term. Third term is sparsity term which is Kullback-Leibler (KL) divergence between two bernoulli random variables with mean p and p_j respectively. It is given as:

$$KL(p\|p_j) = p \log\left(\frac{p}{p_j}\right) + (1-p) \log\left(\frac{1-p}{1-p_j}\right) \quad (3)$$

It measures the difference between two distributions and tries to minimize that difference. Here p is the sparsity parameter set to value very close to 0 and p_j is average activation of hidden unit j averaged over whole training set. This function has the property that it is equal to 0 if $p_j = p$ and otherwise it increases monotonically as p deviates from p_j . $\lambda_{sparsity}$ is sparsity controlling parameter which controls the relative importance of sparsity term in cost function.

Code is written using python programming language to implement sparse autoencoder and its performance is verified using built-in sparse autoencoder in theanets [24] package for neural network which has support for both CPU and GPU. In our code, we trained sparse autoencoder with 420 input nodes, 1680 hidden nodes, 420 output nodes and using 16869 unlabeled examples. Cost function is optimized using gradient based L-BFGS algorithm. Bias terms b_i are initialized to zero while weights w_{ij} are initialized to random numbers drawn from interval $[-\sqrt{\frac{6}{i+o+1}}, \sqrt{\frac{6}{i+o+1}}]$ where i is number of inputs to a node and o is number of outputs from a node [17]. Desired average activation of hidden units $p = 0.01$, sparsity controlling parameter $\lambda_{sparsity} = 0.00003$ and regularization parameter $\lambda_w = 0.003$ have produced best results. After training sparse autoencoder, we forward propagated the labeled data to represent it as activation of hidden units and used that new data as features to SSGO based regressor to predict ASA.

D. Dictionary Learning

Dictionary learning [18] is another method for modeling sparse representations of data. In this method any n -dimensional input vector x is represented as a linear combination $x \approx Dw$ of m -dimensional codes defined in a dictionary $D = [d_1, d_2, \dots, d_m]$. While computing dictionary an additional constraint is imposed that weight vector w is sparse. As w is sparse it means x can only be represented as linear combination of few codes from dictionary. So the problem can be formulated as follows: Given k data points each of n -dimensions represented as $X = [x_1, x_2, \dots, x_k]_{n \times k}$, we want to extract an m -dimensional dictionary $D = [d_1, d_2, \dots, d_m]_{n \times m}$ along with sparse weights $W = [w_1, w_2, \dots, w_k]_{m \times k}$ from input data points. Purpose of this dictionary formulation is to represent each input data point x_i as $x_i = Dw_i$ where w_i is a sparse weight vector. So the cost function is

$$\min_{D, W} \frac{1}{2} \|X - DW\|^2 + \alpha \|W\|_1 \\ \text{s.t. } \|d_i\|_2^2 \leq 1 \forall i = 1, 2, \dots, m.$$

Built-in dictionary learning module of Scikit learn package [25] is used for ASA prediction. First we computed dictionary of size 420×1680 using unlabeled data of size 420×16869 . We have total of 63942 examples of labeled data. After dictionary is computed, labeled data is transformed using this pre-computed dictionary. Transformed data is of dimensions 1680×63942 . Only tuning parameter is regularization parameter α which is set to be 0.05 using grid search method. Transformed data is fed as feature vectors to SSGO based regressor to predict accessible surface area of proteins.

E. Principal Component Analysis

Principal component analysis (PCA) is a data modeling technique to reduce dimensions of data. It is also used to rotate data to achieve simple structure of data which is useful from classification or regression point of view. As it reduces the dimensions as well as redundancy in data, so it increases speed as well as accuracy for unsupervised feature learning algorithms. Suppose you are training your algorithm on some kind of data which has redundancy due to correlation among adjacent data points PCA will help to get a better representation of data which is lower dimensional with very little error as only redundant dimensions are removed.

First step of PCA is to merge all data points or feature vectors in a single matrix as class labels are not required for PCA. Then compute the n -dimensional mean vector where n is the dimension of each feature vector.

$$m = \frac{1}{k} \sum_{i=1}^k x_i \quad (4)$$

Next step is to compute the co-variance matrix using following equation:

$$C = \frac{1}{k-1} \sum_{i=1}^k (x_i - m)(x_i - m)^T \quad (5)$$

Then compute the eigen vectors and corresponding eigen values of co-variance matrix. Next step is to sort the eigen values from highest to lowest and choose the eigen vectors

corresponding to top p eigen values. Eigen values define the length of eigen vectors or amount of variance in the direction of corresponding eigen vectors. Place the p selected eigen vectors into projection matrix T . Transform original n -dimensional data matrix X into new m -dimensional subspace Y using $Y = W^T X$.

We have implemented PCA using Scikit learn package. Input data matrix consists of 63942 data points with each of dimension 420. Dimensionality of input data points is preserved as all the eigen values are very small and close with one another. We have selected all the 420 eigen vectors and transform our data to new subspace using this projection matrix. The projected data is used as feature vectors to SSGO based regressor and ASA of proteins is predicted.

F. Deep Neural Network

Deep learning is a new area of machine learning with purpose of learning higher level abstractions in data using complex structures or multiple nonlinear transformations. One of the simplest example of deep learning is multilayer perceptron or feedforward neural network with multiple layers [21]. In a deep neural network, each layer trains on a distinct set of features based on the previous layers output. The further you advance into the network, the more complex and improved features your nodes can recognize, since the features learned from the previous layers are used as input to next layers. These algorithms can be both supervised and unsupervised. For supervised learning tasks, deep learning algorithms learn better representation of data reducing redundancy in data.

We have implemented a deep neural network based regressor using Theanets package for neural networks. Only labeled data is used to train the network. Network consists of three hidden layers each having 1500 nodes, input layer having 420 nodes and output layer has single node. Learning algorithm used is rmsprop which is gradient decent based algorithm with adaptive learning rate. Regularization parameter and learning rate are set to 0.01 and 0.0001 respectively. All of these parameters are selected using some rules of thumb available in literature and grid search.

G. Support vector Regression

ASA prediction problem is formulated as a support vector regression problem. Each residue in the proteins in the training set is encoded into a feature vector x_i . Then, x_i is mapped (non-linearly) onto an m -dimensional feature space. A linear model is constructed in this feature space. The predicted ASA value will be given by

$$f(x) = \sum_{k=1}^n w_k \Phi(x) + b \quad (6)$$

Here, $\Phi(x)$ is the non-linear mapping of feature vectors and b is the bias. The regression parameters w_j and b are estimated by minimizing the norm of the weights, $\|w\|^2$, and the empirical risk function on the training samples. In particular Vapnik's ϵ -insensitive loss function is used here to minimize the errors.

It is defined by

$$L_\epsilon(y - f(x)) = \begin{cases} 0, & \text{if } |(y - f(x))| \leq \epsilon \\ |(y - f(x))| - \epsilon, & \text{otherwise} \end{cases} \quad (7)$$

This term sets the tolerance to error in loss function. Errors smaller than ϵ are not considered as errors. So overall risk function to estimate w_j has two terms given as:

$$\frac{1}{2} \|w\|^2 + \frac{C}{n} \sum_{k=1}^n L_\epsilon(y_k - f(x_k)) \quad (8)$$

Where C is a user-settable regularization constant and n is the total number of training examples. Regularization constant is to set trade-off between prediction error and complexity of model. Smaller value of C will simplify the model while larger value of C will cause overfitting.

Problem can be transformed into constrained optimization problem by using slack variables. This constrained optimization problem is solved by adding Langrange multipliers and dual form of the problem is developed. If data is not linearly separable, it can be transformed to some higher dimensional space using some nonlinear kernel function. We have tried Radial Basis Function (RBF) to map input data to some higher dimensional feature space. RBF kernel between two input features x_i and x_j is defined as:

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2) \quad (9)$$

Where parameter γ is user settable and it defines the spread of data. Dual form of above mentioned problem is solved by using quadratic programming solver [19]. It is implemented using Scikit-learn. Scikit-learn is simple and efficient toolbox for Machine learning in Python.

H. Stochastic Subgradient Optimization based Regressor

Quadratic optimization problem formulated above is time consuming for larger datasets so stochastic sub-gradient descent algorithm [20] is used which is simple and effective for solving the optimization problem cast by Support Vector Machines (SVM). It has an edge over other convex optimization algorithms that its time complexity is linear and independent of number of data points [20]. For linear kernels, the total run-time of this algorithm is $O(\frac{d}{\lambda \epsilon})$, where d is a bound on the number of non-zero features in each example and λ is the regularization parameter of SVM. Input to this algorithm is data values, target values, number of iterations and regularization constant while it outputs parameters of regressor. On each iteration it operates as follow. Initially, we set weight vector w to the zero vector. On iteration t of the algorithm, we first choose a random training example (x_{it}, y_{it}) by picking an index $i_t \in \{1, 2, \dots, m\}$ uniformly at random. Sub-gradient of objective function in expression 8 at chosen example is evaluated and added to previous value of w to update it for next iteration.

Pseudo code for linear form of SSGO algorithm is shown in Algorithm 1.

Where S is set of data and target values, λ is regularization constant, T is number of iterations and $\eta = \frac{1}{\lambda T}$ is called learning rate. It is implemented using numpy package for array operations of Python programming language.

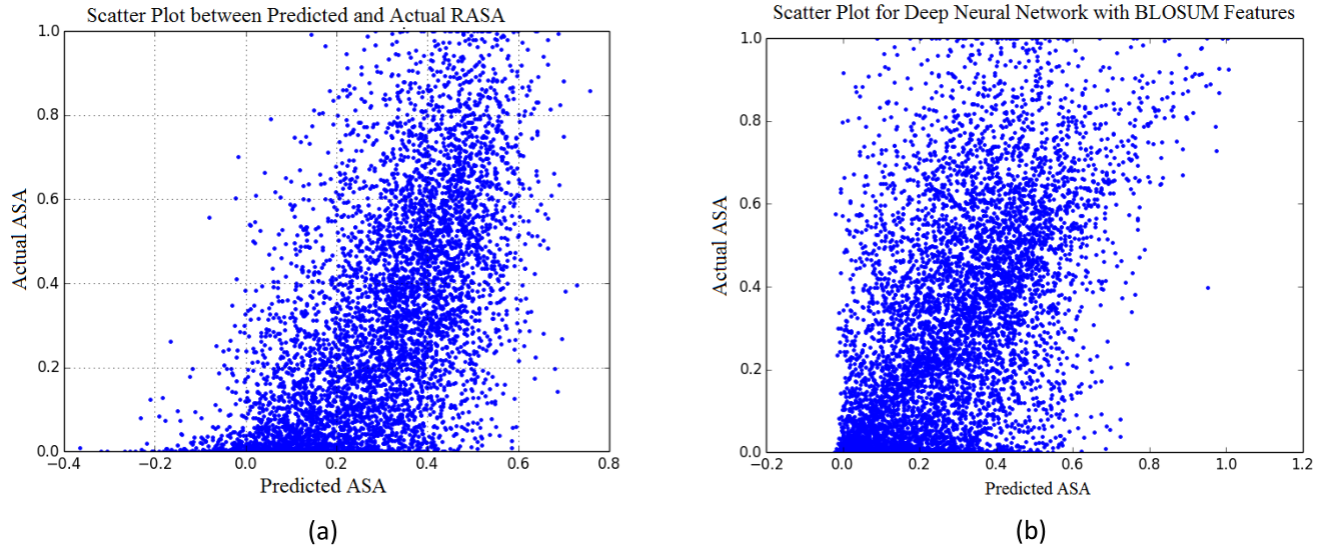


Fig. 1. (a) Scatter plot between actual and predicted ASA using PSSM based features and SSGO algorithm (b) Scatter plot between actual and predicted ASA for deep neural network with Blosum 62 based features

Algorithm 1: Stochastic Sub-gradient Optimization Algorithm

Input: S, λ, T

- 1 INITIALIZE $W_1 = 0$
- 2 **for** $t = 1, 2, \dots, T$ **do**
- 3 Choose $i_t \in \{1, 2, \dots, |S|\}$ uniformly at random
- 4 SET $\eta_t = \frac{1}{\lambda t}$
- 5 **if** $\langle W_t, X_{it} \rangle - y_{it} > \epsilon$ **then**
- 6 $W_{t+1} = (1 - \eta\lambda)W_t - \eta X_{it};$
- 7 **else if** $y_{it} - \langle W_t, X_{it} \rangle > \epsilon$ **then**
- 8 $W_{t+1} = (1 - \eta\lambda)W_t + \eta X_{it};$
- 9 **else**
- 10 $W_{t+1} = (1 - \eta\lambda)W_t$
- 11 **end**
- 12 **end**

III. RESULTS AND DISCUSSION

After cleaning PDB files and removing files shorter than 30 residues length, we were left with a dataset of 63942 examples from 343 unbound protein chains. The parameter of Vapnik's ϵ -insensitive loss function was set as 0.01 for SVR. RBF kernel is used and various values of C and γ were tried and got best results for $C = 0.1$ and $\gamma = 0.1$. As quadratic programming based solver was much slower, so we used stochastic sub-gradient algorithm which is computationally much faster as compared to first method and gave prediction accuracy comparable with quadratic programming based solver. Scatter plot between predicted and actual RASA is shown in Figure 1(a) for PSSM based features. All the results using two types of optimization algorithm, three types of features and three self-taught learning algorithms are mentioned in Table I. Among the three types of features used, PSSM based features are superior features as they include more information of presence of a residue at a particular location as well as neighborhood information and almost whole of the protein universe is traversed to compute these features.

It took about 10 hours to optimize dual form of SVR using quadratic programming based solver whereas stochastic sub-gradient based regressor took only 10 minutes to optimize the basic optimization problem of SVR. We have tried linear, polynomial and RBF kernel for both methods. First method produced best results using RBF kernel while second method linearly solved the problem and produced comparable results. So SSGO regressor with PSSM features is the finding of this work. Pearson's correlation coefficient obtained for all

Input	SSGO	QP Solver	Autoencoder	Dictionary	PCA	Deep NN
			SSGO	SSGO	SSGO	
PD	0.50	0.54	0.50	0.49	0.50	0.49
BLOSUM62	0.50	0.53	0.49	0.49	0.50	0.58
PSSM	0.63	0.64	--	--	--	--

TABLE I. PREDICTION RESULTS SHOWING PEARSON'S CORRELATION COEFFICIENT BETWEEN PREDICTED AND TRUE ASA VALUES USING 10-FOLD CROSS VALIDATION

self-taught learning algorithms is also presented in Table I. Results are same using all self-taught learning algorithms. Self-taught learning algorithms have shown comparable results with methods without these algorithms which is an indication that data is already sparse and there is no redundancy in the data. Our objective was to achieve prediction accuracy comparable with PSSM based features using Blosum62 and position dependent k-spectrum based features. Deep neural network with Blosum62 features has shown improvement in prediction accuracy as these features are based on similarity information of residues. Position dependent k-spectrum has not shown any improvement because of its sensitivity to position of residues in protein sequence. Shifting of residues in sequence causes significant changes in these features. PSSM based features are much superior as compared to other two features as it requires traversal of almost whole of protein universe to compute these features. It requires lot of data and time to compute these features so we have not considered these features for self-taught or deep learning implementation. On

other hand, Blosum62 features are simple features and easy to compute. So the improvement in prediction accuracy with Blosum62 based features is mainly due to the use of deep neural network. Scatter plot between predicted and actual ASA values for deep neural network is shown in Figure 1(b).

IV. CONCLUSION & FUTURE WORK

We have used quadratic programming to solve dual form of kernelized SVR to predict ASA which is time consuming but use of SSGO algorithm for prediction of ASA has given comparable results with first method with much lesser time complexity. SSGO algorithm has an advantage over other methods that its computation complexity does not depend upon number of data points rather it depends upon dimensions of feature vectors and the learning rate used. It has been shown that using Position Specific Scoring Matrix (PSSM) as feature has significantly improved the prediction accuracy of SVR as compared to other two types of features. In the end we can conclude that using SSGO based regressor along with PSSM based features is time efficient and reasonably accurate method to cope up the problem of big data for ASA prediction of proteins. One very important finding of this research work is use of deep learning and self-taught learning algorithms to predict ASA of proteins as a large amount of unlabeled data of proteins is available. It will also solve the problem of suitable feature selection stage for prediction of protein properties.

Using self-taught learning algorithms, we got same results as with simple SSGO regressor but one very important finding of the work is use of unlabeled data to learn protein universe. Large amount of unlabeled data is available in the form of protein sequences. According to statistics, about 30 million protein sequences are available in non redundant database but structural information of only 1,12,561 proteins is available [26]. It can urge researchers to use these algorithms and unlabeled data to solve problems in Bioinformatics. In future these algorithms can be used to predict multiple properties simultaneously as it has shown improvement in performance of predictors using deep neural networks [13] [15] [14].

REFERENCES

- [1] Joseph A. Marsh, "Buried and Accessible Surface Area Control Intrinsic Protein Flexibility," *Journal of Molecular Biology*, vol. 425, no. 17, pp. 3250–3263, Sept. 2013.
- [2] Song Liu, Chi Zhang, Shide Liang, and Yaoqi Zhou, "Fold recognition by concurrent use of solvent accessibility and residue depth," *Proteins: Structure, Function, and Bioinformatics*, vol. 68, no. 3, pp. 636–645, 2007.
- [3] Jun-Feng Xia, Xing-Ming Zhao, Jiangning Song, and De-Shuang Huang, "Apis: accurate prediction of hot spots in protein interfaces by combining protrusion index with solvent accessibility," *BMC bioinformatics*, vol. 11, no. 1, pp. 174, 2010.
- [4] Zheng Yuan, Kevin Burrage, and John S. Mattick, "Prediction of protein solvent accessibility using support vector machines," *Proteins: Structure, Function, and Bioinformatics*, vol. 48, no. 3, pp. 566–570, Aug. 2002.
- [5] Minh N. Nguyen and Jagath C. Rajapakse, "Prediction of Protein Relative Solvent Accessibility with a Two-Stage SVM," *Approach, Proteins: Structure, Function, and Bioinformatics*, pp. 30–37, 2005.
- [6] Stephen R Holbrook, Steven M Muskal, and Sung-Hou Kim, "Predicting surface exposure of amino acids from protein sequence," *Protein Engineering*, vol. 3, no. 8, pp. 659–665, 1990.
- [7] Gianluca Pollastri, Pierre Baldi, Pietro Fariselli, and Rita Casadio, "Prediction of coordination number and relative solvent accessibility in proteins," *Proteins: Structure, Function, and Bioinformatics*, vol. 47, no. 2, pp. 142–153, 2002.
- [8] Shandar Ahmad, M. Michael Gromiha, and Akinori Sarai, "Real value prediction of solvent accessibility from amino acid sequence," *Proteins: Structure, Function, and Bioinformatics*, vol. 50, no. 4, pp. 629–635, Feb. 2003.
- [9] Zheng Yuan and Bixing Huang, "Prediction of protein accessible surface areas by support vector regression," *Proteins*, vol. 57, no. 3, pp. 558–564, Nov. 2004.
- [10] Zheng Yuan and Timothy L. Bailey, "Prediction of protein solvent profile using SVR," *Conference proceedings: ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference*, vol. 4, pp. 2889–2892, 2004.
- [11] Rafa Adamczak, Aleksey Porollo, and Jaroslaw Meller, "Accurate prediction of solvent accessibility using neural network-based regression," *Proteins: Structure, Function, and Bioinformatics*, vol. 56, no. 4, pp. 753–767, Sept. 2004.
- [12] Aarti Garg, Harpreet Kaur, and GPS Raghava, "Real value prediction of solvent accessibility in proteins using multiple sequence alignment and secondary structure," *Proteins: Structure, Function, and Bioinformatics*, vol. 61, no. 2, pp. 318–324, 2005.
- [13] Rafał Adamczak, Aleksey Porollo, and Jarosław Meller, "Combining prediction of secondary structure and solvent accessibility in proteins," *Proteins: Structure, Function, and Bioinformatics*, vol. 59, no. 3, pp. 467–475, 2005.
- [14] Yanjun Qi, Merja Oja, Jason Weston, and William Stafford Noble, "A Unified Multitask Architecture for Predicting Local Protein Properties," *PLoS ONE*, vol. 7, no. 3, pp. e32235, Mar. 2012.
- [15] Rhys Heffernan, Kuldip Paliwal, James Lyons, Abdollah Dehzangi, Alok Sharma, Jihua Wang, Abdul Sattar, Yuedong Yang, and Yaoqi Zhou, "Improving prediction of secondary structure, local backbone angles, and solvent accessible surface area of proteins by iterative deep learning," *Scientific reports*, vol. 5, 2015.
- [16] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng, "Self-taught learning: transfer learning from unlabeled data," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007.
- [17] Andrew Ng, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, 2011.
- [18] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009.
- [19] Chih-Chung Chang and Chih-Jen Lin, "Libsvm: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, pp. 27, 2011.
- [20] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter, "Pegasos: primal estimated sub-gradient solver for SVM," *Mathematical Programming*, vol. 127, no. 1, pp. 3–30, Oct. 2010.
- [21] Yoshua Bengio, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.
- [22] Howook Hwang, Thom Vreven, Jol Janin, and Zhiping Weng, "Protein-Protein Docking Benchmark Version 4.0," *Proteins*, vol. 78, no. 15, pp. 3111–3114, Nov. 2010.
- [23] Matthias Heinig and Dmitriy Frishman, "STRIDE: a web server for secondary structure assignment from known atomic coordinates of proteins," *Nucleic Acids Research*, vol. 32, no. Web Server issue, pp. W500–502, July 2004.
- [24] Leif Johnson, "Theanets 0.6.2," 2015.
- [25] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al., "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [26] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, TN Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne, "The protein data bank," *Nucleic acids research*, vol. 28, no. 1, pp. 235–242, 2000.